Project Work
Combinatorial Decision Making & Optimization 2022/2023

Topic: Modelling & solving the multiple couriers planning problem

Authors: Zeynep Kiziltan and Roberto Amadini

Date: October 27, 2023

This document describes the project work of the Combinatorial Decision Making and Optimization course for the academic year 2022/2023, which is about modelling and solving the multiple couriers planning problem. The students can propose another problem from the literature provided that they get our approval before start working on it. The project work involves approaching the problem using (i) Constraint Programming (CP), (ii) propositional SATisfiability (SAT) and/or its extension to Satisfiability Modulo Theories (SMT), and (iii) Mixed-Integer Linear Programming (MIP).

The students **must form a group** of 2-3 members, and it suffices for point (ii) to develop a SAT or an SMT solution. If they do both, they obtain bonus points. The students can also form a group of 4 members, but in that case they need to develop both SAT and SMT solutions. A student who cannot form a group must get our approval with a convincing motivation.

1 Problem description

It has become very common recently, especially after the outbreak of the COVID-19 pandemic, to send or receive items online (food, goods, clothes, documents, ...). Hence, it is more and more important for companies to efficiently schedule the dispatching of items through different couriers, in order to minimize the transportation costs.

As the combinatorial decision and optimization expert, students are asked to solve the Multiple Couriers Planning (MCP) problem which is defined as follows. We have m couriers that must distribute $n \geq m$ items at different customer locations. Each courier i has a maximum *load size* l_i. Each item j has a *distribution point* j and a *size* s_j (which can represent for instance a weight or a volume). The goal of MCP is to decide for each courier the items to be distributed and plan a *tour* (i.e. a sequence of location points to visit) to perform the necessary distribution tasks. Each courier tour must start and end at a given origin point o. Moreover, the maximum load l_i of the courier i should be respected when items are assigned to it. To achieve a fair division among drivers, the objective is to minimize the maximum distance travelled by any courier.

Instance format An instance of MCP is a text file of the form:
m
n
$l_1 \; l_2 \; \dots \; l_m$
$s_1 \; s_2 \; \dots \; s_n$
$D_{1,1} \; D_{1,2} \; \dots \; D_{1,n+1}$
$D_{2,1} \; D_{2,2} \; \dots \; D_{1,n+1}$
$\dots$
$D_{n,1} \; D_{n,2} \; \dots \; D_{n,n+1}$
$D_{n+1,1} \; D_{n+1,2} \; \dots \; D_{n+1,n+1}$
where m, n, l_i, s_j have the meaning explained above, while $D_{i,j}$ is the distance between distribution point i and distribution point j for $i, j = 1, \dots, n+1$. In particular, distribution point $n+1$ corresponds to the origin origin point (clearly, $D_{i,i} = 0$ for $i = 1, \dots, n+1$). For example, the following input file:

```
3
7
15 10 7
3 2 6 8 5 4 4
0 3 3 6 5 6 6 2
3 0 4 3 4 7 7 3
3 4 0 7 6 3 5 3
6 3 7 0 3 6 6 4
5 4 6 3 0 3 3 3
6 7 3 6 3 0 2 4
6 7 5 6 3 2 0 4
2 3 3 4 3 4 4 0
```

corresponds to a MCP instance with 3 couriers and 7 items where, e.g., $D_{2,n+1} = D_{2,8} = 3$ is the distance between the distribution point 2 to o, while $D_{4,7} = 6$ is the distance between the distribution points 4 and 7. Note that in general D is *not* guaranteed to be symmetrical, i.e., it is possible that $D_{i,j} \neq D_{j,i}$. The optimal solution to this instance is depicted in Fig. 1, where d_i refers to the distribution point i and the distance matrix D corresponds to the Manhattan distance between the locations of the various points.

2 Project work

The purpose of the project work is to model and solve the MCP problem using (i) Constraint Programming (CP), (ii) propositional SATisfiability (SAT) and/or its extension to Satisfiability Modulo Theories (SMT), and (iii) Mixed-Integer Linear Programming (MIP). The students will decide themselves the decision variables and the problem constraints around the problem description. The project work includes building models as well as conducting an experimental study to assess the performance of the solvers using different models and search strategies (where applicable). A suite of problem instances will be provided in

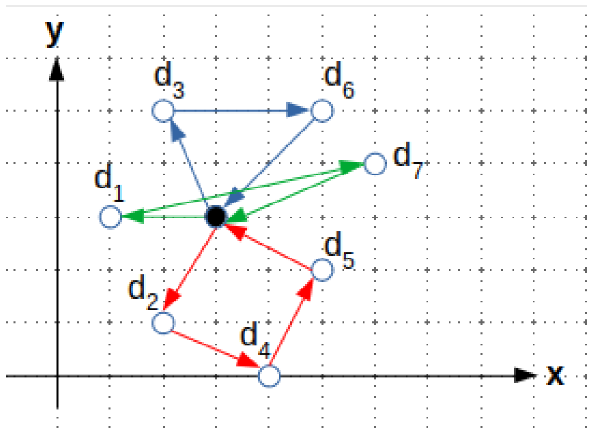

Figure 1: Graphical representation of an optimal MCP solution. The black node is the origin point, the white nodes are the distribution points and the arrows denote the tour of each courier.

the format specified in Section 1 for experimental purposes. No assumptions should be made based on these instances. The approach should work for any random instance.

2.1 Project groups

The students **must form a group** of 2-3 members, and it suffices for point (ii) to develop a SAT or an SMT solution. If they do both, they obtain bonus points. The students can also form a group of 4 members, but in that case they need to develop both SAT and SMT solutions.

The names of the group members and their e-mail addresses must be indicated in the following sheet before project submission: `shorturl.at/iEF27`. The sheet can be used also to write down individual names in search for partners and to form groups. A student who cannot form a group must get our approval with a convincing motivation before start working on the project.

2.2 Project development

When developing the project, students should pay attention to the following:

- Start with the instance parameters, decision variables, domains and the objective function.
- Constrain the objective function with tight lower and upper bounds.
- Though a trivial model is a good starting point, focus on the peculiarities of the underlying optimization approach and try to come up with the best model so as to be able to tackle the instances in the best possible ways.

- Investigate the best way to search (where applicable).
- The CP model **must** be implemented in MiniZinc and run using at least Gecode, whereas the SAT model **must** be implemented using at least Z3. For both SMT and MIP models, students can use their favourite theories, solvers and languages. Bonus points are considered if the problem is modelled with a solver-independent language, so as to try different solvers without rewriting the model. MiniZinc should be avoided here: models automatically derived from a MiniZinc model will not be considered. Students are welcome to exploit different solver parameter configurations.
- Solving processes that exceed a time limit of 5 minutes (300 secs) should be aborted. Depending on the instances, on the solving approach, on the machine, etc., some of the instances may be too difficult to solve within the time limit. For this reason, it is not strictly necessary to succeed in solving all instances to pass the project. However, students are encouraged to improve the solution quality obtained by the time limit by using best practices.
- If any *pre-solving* technique is used before invoking a solver (e.g., sorting or rearranging items) then include the time taken for pre-solving in the reported total runtime.

2.3 Project report

The students should describe clearly their work in a report. While a MiniZinc notation could be acceptable to describe a CP model, it is required in general to use propositional logic, first-order logic and appropriate mathematical notations. **Do not copy and paste code!** The report should also describe the experimental results obtained by the provided instances by focusing on the best combination of model and search (where applicable).

It is mandatory to produce one single report following the template prepared by the course teachers. The report should be written in LaTeXusing `\documentclass{article}` without changing fonts, font size and margins. The page limit is 12 pages with three optimization approaches and 15 pages with all the approaches, excluding the references. You are recommended to use `www.overleaf.com/` to produce a shared LaTeXdocument.

2.4 Solution format

Collect all the solutions under a `res` folder, by creating specific sub-folders (e.g. `res/SAT`, `res/SMT`, `res/MIP`, etc.) to store the results of each optimization approach. For all the instances solved by a given model, you must present a corresponding `.json` file. For instance, the results of your MIP model on instance #3 should be placed under the path `res/MIP/3.json`. The keys of each `.json` file are the names of the approaches presented in the experimental results of the report. Use meaningful names and be coherent with what you present in the report. Each approach must be documented with its *runtime* (`time`, an integer), *optimality state* (`optimal`, a Boolean true iff the instance is solved to optimality), *objective function value* (`obj`, a positive integer), and the actual *solution* (`sol`, a list of integer lists). For the `time` field, use the *floor* of the actual runtime (e.g., if the actual runtime is 42.7s then `time` = 42). If an approach unexpectedly terminates before the timeout (300s) without solving to optimality, then set the `time` field to 300. In this way, `time` = 300 $\iff$ `optimal` = $false$. The `sol` field must be a list of fixed length m where, for $i = 1, \dots, m$, `sol`$[i]$ is the *ordered* list of all the items collected by courier i (hence, `sol`$[i][j]$ is the j-th item collected by courier i). The order of couriers and items is the one given in the input file. For example:

```
{
    "gecode":
    {
        "time": 300,
        "optimal": false,
        "obj": 12,
        "sol" : [[3, 6, 5], [4, 2], [7, 1]]
    },
    "gecode_symbreak":
    {
        "time": 120,
        "optimal": true,
        "obj": 12,
        "sol" : [[3, 6, 5], [4, 2], [7, 1]]
    },
    "chuffed":
    {
        "time": 300,
        "optimal": false,
        "obj": 22,
        "sol" : [[5, 3, 7], [2, 4], [6, 1]]
    }
}
```

2.5 Project submission

The project will be submitted via Virtuale, and it shall contain the report (as described in Section 2.4), the results (as described in Section 2.3) and the source code. To be reproducible, the source code must be executable through *Docker*. Inside the code folder there must be the Dockerfile to build the *Docker* image, along with the instructions to run each of your models. The instructions should be concise and precise to allow the reproduction of the results. You can either submit an `.sh` script which takes as input an instance number and runs the model on that instance or provide a README file. Keep in mind that the solvers need to be executed inside the Docker container defined by your Dockerfile, and not just in your local machine. Therefore, it is strongly suggested to use free software only. The choices for the instructions format and the implementation of the *Docker* environment are completely up to you.

ATTENTION All the sources needed to reproduce the experiments (e.g., `.mzn`, `.dzn`, `.smt2` files) must be uploaded. The correctness of the computed solutions must be checked with the solution checker `check_solution.py` available on virtuale webpage. Any submission where the `.json` files containing the solutions:

- **cannot be processed by the checker, or**
- **contain errors reported by the checker**

will be automatically rejected.

Name your submission folder as `CDMO_Proj_LastnameName1_..._LastnameNameK` and upload its `tgz` or `zip` compressed archive. When working as a group, it suffices that only one group member submits the project, provided that the following information is provided in the **submission comments**:

- all the members' names, lastnames and UniBo e-mail addresses;
- the impossible dates and time for discussion due to a hard constraint (like another exam, medical appointment, employment, urgent family matter).

3 Evaluation, examination and grading

ATTENTION if you have any specific deadline by which you need to register the mark (for instance due to a scholarship, graduation etc), please contact the course teachers well in advance! We cannot guarantee to meet personal deadlines which have not been communicated earlier.

While it suffices to produce either a SAT or an SMT encoding for groups of max 3 students, bonus points will be given to those who produce both. Groups of 4 students are required to produce both.

As part of the project evaluation, group members will be invited for an oral discussion of the project (submitted at an earlier date) with one of the course teachers. No additional presentation is required. At each exam session, there will be a single date for submission and multiple dates for discussions. The discussion schedule will be devised taking into account students' hard constraints and will be communicated some days after the submission.

Important dates

- July 2023: Submission 09/07, Discussions 13/07, 14/07, 17/07
- September 2023: Submission 03/09, Discussions 11/09, 12/09, 13/09

There will be other exam sessions in December 2023 and February 2024. Those who have not yet discussed by February 2024 will be able to submit later with the risk of having to wait until the summer of 2024 for discussion. Discussions will take place online on Teams.

The final evaluation will be based on the students' ability to model and solve a combinatorial optimization problem using the approaches and the tools discussed in the course, as well as the ability to present their work. Particular attention will be given to the literature search, originality, technical quality, and clarity of presentation.

A successful evaluation will result in a mark between 18 and 30L. **ATTENTION Students on a mobility leave (Eramus Overseas, etc) during the discussion can register their marks only after their return to UniBo.** An insufficient project will not obtain a mark. In that case, the group members will be asked to rework on the project and resubmit.

A group may decide to reject a mark proposed by the teacher and resubmit the project *only once*. In that case, no new members can be added to the project. Note that the final mark may be better, the same or even worse than the original mark depending on how satisfactory/significant the revision is.

4 Academic integrity

Intellectual development requires honesty, responsibility, and doing your own work. Plagiarism is the use of someone else's work, words, or ideas as if they were your own. Any idea taken from any person or resource should be cited appropriately. Plagiarised work will not be evaluated.